Reconfigurable Logic Embedded Architecture of Support Vector Machine Linear Kernel

Jeevan Sirkunan*, N. Shaikh-Husin†, Trias Andromeda‡, and M. N. Marsono§
*†§Fac. of Electrical Eng., Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
‡Department of Electrical Engineering, Diponegoro University, Semarang, Indonesia, 50275.
*jeevan2@live.utm.my, †nasirsh@fke.utm.my, ‡triasandromeda@undip.ac.id, §nadzir@fke.utm.my

***Abstract*—Support Vector Machine (SVM) is a linear binary classifier that requires a kernel function to handle non-linear problems. Most previous SVM implementations for embedded systems in literature were built targeting a certain application; where analyses were done through comparison with software implementations only. The impact of different application datasets towards SVM hardware performance were not analyzed. In this work, we propose a parameterizable linear kernel architecture that is fully pipelined. It is prototyped and analyzed on Altera Cyclone IV platform and results are verified with equivalent software model. Further analysis is done on determining the effect of the number of features and support vectors on the performance of the hardware architecture. From our proposed linear kernel implementation, the number of features determine the maximum operating frequency and amount of logic resource utilization, whereas the number of support vectors determines the amount of on-chip memory usage and also the throughput of the system.**

I. Introduction

Future server and embedded systems will likely to consist of built-in intelligent machine learning classifiers [1]. These applications need to process large amounts of data, and this requires massive data parallelism that needs high data bandwidth between the processors and off-chip memory. Such data access patterns make on-chip caches mostly ineffective [1]. Reconfigurable hardware such as field programmable gate array (FPGA) is a promising platform for speeding up computations, and for providing high performance computing (HPC) at minimal cost overhead and power consumption [2].

SVM is an accurate classifiers that is based on a solid theoretical background for hardware implementation [3], [4]. Some applications that have benefited from SVM hardware acceleration are pedestrian detection [5], face detection [6] and object detection [7]. SVM has also been deployed for medical application such as cardiac arrhythmia detection [8]. Many existing custom hardware implementations focused on accelerating the decision function that is application specific [6], [9]. These designs cannot be easily reused for other applications. Besides that, these design are targeted for fixed number of classes [6].

Works that targeted on accelerating SVM with a co-processor unit [1], [10]–[12] tend to focus on the kernel due to its compute-intensive task and also its innate nature to be parallelize. Prior work by Kane et al. [13] implemented a generic SVM classification architecture that was tested with a wide variety of datasets. However, the analysis is only limited to comparison with software implementations only. The effect of different applications on SVM was not reported such as the number of features, support vectors and class that vary from one application dataset to another. In this work, we propose a linear kernel architecture on FPGA based Kane et al. [13] work. The proposed design is analyzed in terms of hardware resource and maximum operating frequency with regards to different number of support vectors and features.

The rest of this paper is organized as follows. Section II discusses the overview of SVM. Section III presents related works in SVM with hardware support. Section IV elaborates the architecture design of the linear kernel implementation. Section V discusses the performance evaluation of the hardware implementation with regards to the number of support vectors and features. This paper is concluded in Section VI with suggestions for future work.

II. Overview of SVM

Traditionally, there have been two fundamentally different types of tasks in machine learning: unsupervised and supervised. The goal of unsupervised learning is to find patterns and structure in the data, while supervised learning is to learn a mapping from a labeled dataset [14]. SVM falls under the category of the latter. In general, SVM task can be divided into two: training and classification. Based on a set of labeled dataset, SVM is trained to determine its decision function. With the decision function, unknown data can then be classified.

A. Training

In the training phases, SVM determines the decision boundary that maximizes the space between two classes [15]. In order to handle data which cannot be separated linearly in low-dimensional feature space, kernel functions are used to project learning data into high-dimensional space so that it can be linearly separated [16]. Eqs. (1) and (2) show the dual Lagrange problem to obtain the SVM decision function. $K(x_i, x_j)$ is the kernel function, b is the bias parameter and C is the trade-off parameter between maximizing the margin and minimizing the error.

$$\text{Maximize } L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(x_i, x_j)$$
$$\text{Subject to} \begin{cases} 0 \leq \alpha \leq C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{1}$$

$$b = y_i - \left(\sum_{i=1}^{N} \alpha_i x_i y_i\right) \tag{2}$$

where :

α :	Lagrangian multiplier
N :	Total amount of training dataset
x_i :	Training data (feature set)
y_i :	Training data (label)

The main aim of the training function (eqs. (1) and (2)) is to obtain α_i and b values. If α_i is 0, the corresponding training feature vector x_i is not a support vector. If α_i is very high, then the corresponding training feature x_i has a high influence over the decision surface of the hyperplane (x_i becomes a support vector).

B. Classification

Eq. (3) shows the SVM decision function $d(u)$ for classification. The computation only extends to the number of support vectors N_{sv}, where the support vectors are the subsets of the training data. Based on the output sign from eq. (3), the class of unknown input of u is determined by

$$d(u) = \text{sign}\left(\sum_{i=1}^{N_{sv}} \alpha_i K(x_i, u) + b\right) \tag{3}$$

C. Kernel Function

Kernel functions efficiently map non-linear datasets to a highly dimensional linear feature space. Kernel allows SVM to handle non-linear data. The type of kernel to be employed is entirely dependent on the characteristics of the applications datasets, which is beyond the scope of this paper. Some of the main kernel functions that are used in literature are linear (eq. (4)) polynomial (eq. (5)) Gaussian radial function (RBF) (eq. (6)) and sigmoid (eq. (7)).

$$K(x_i, u) = x_i \cdot u \tag{4}$$
$$K(x_i, u) = (\alpha(x_i \cdot u) + r)^d \tag{5}$$
$$K(x_i, u) = e^{(-\gamma)\|x_i - u\|^2}, \text{where } \gamma > 0 \tag{6}$$
$$K(x_i, u) = \tanh(\alpha(x_i \cdot u) + r) \tag{7}$$

III. Related Works

Extant works on accelerating SVM on FPGA can be divided into two main groups: training and classification. Accelerating training phase focused for certain applications where on-line learning is required. These applications, e.g. [17], [18] demand short training time for fast adaptation with new user habits and since the characteristics of data change over time. On the other hand, accelerating classification phase is important when a task requires a fixed classification module with large data to be classified [6], [9]. Depending on the application requirement, either the training or classification are implemented on FPGA.

For optimizing SVM training problem, SMO [19] is considered as state-of-the-art training algorithm. However, the SMO algorithm itself was designed in such away that it caters single-threaded computer [12]. Besides SMO, there are also Gilbert's algorithm [20] and LS-SVM [21] for training. Works that are targeted for training either implements the whole system on-chip or offloading to a co-processor [1], [10]–[12] to accelerate the training process. The task that is targeted for hardware implementation is the kernel as it is compute-intensive task that benefit from parallelization.

For the classification part, there were numerous acceleration works on FPGA. Many existing custom hardware implementations focused on accelerating the decision function for a specific task [6], [9]. These designs cannot easily be reused for other purposes as these designs were targeted for fixed number of classes. Another approach toward SVM classification on FPGA system is the cascaded SVM [22]. Cascaded SVM architecture contains two modules with varying precision rate that interchange depending on the testing data. However in this work [22], the focus was solely on binary classification problem.

There were certain works that targeted multi-class classification and tested on various number of datasets [23]. However, most works in literature were not benchmarked with CPU or GPU implementation. Kane et al. [13] proposed a SVM decision architecture with a generic design which was then tested with a wide variety of datasets. This work outperformed GPU and CPU implementations in terms of speed with practically a negligible trade-off in accuracy. However, the impact on different types of datasets towards the architecture's hardware resource and performance were not reported. In embedded system environment, hardware resources such as on-chip memory or logic resource are limited. In this work, we will analyze factors that affect hardware resource utilization and performance in implementing SVM linear kernel on FPGA.

IV. Linear Kernel Embedded Architecture

In this section, we describe the proposed linear kernel architecture in this paper. The general aim is to make the architecture parameterizable so it can be optimized in terms of hardware utilization depending on task requirement. Eq. (8) shows the computations that take place in the linear kernel. $x_{i,j}$ are the testing data, $SV_{i,j}$ are the support vectors, N_{sv} is the total number of support vector and N_f is the total number of features. Each test data needs to perform dot product with each of the support vector.

Fig. 1 shows the architecture of the linear kernel. The linear kernel architecture is based on the work from Kane et al. [13]. Due to its fully pipelined design, after an initial latency, the architecture produces output in every clock cycle. In our implementation, all storage memories are implemented using

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,N_f} \end{bmatrix} \begin{bmatrix} SV_{1,1} & SV_{1,2} & \dots & SV_{1,N_f} \\ SV_{2,1} & SV_{2,2} & \dots & SV_{2,N_f} \\ \vdots & \vdots & \vdots & \vdots \\ SV_{N_{sv},1} & SV_{N_{sv},2} & \dots & SV_{N_{SV},N_f} \end{bmatrix} = \begin{bmatrix} x_{1,1} \cdot SV_{1,1} + x_{1,2} \cdot SV_{1,2} \cdots + x_{1,N_f} \cdot SV_{1,N_f} \\ \vdots \\ x_{1,1} \cdot SV_{N_{sv},1} + x_{1,2} \cdot SV_{N_{sv},2} \cdots + x_{1,N_f} \cdot SV_{N_{SV},N_f} \end{bmatrix} \tag{8}$$

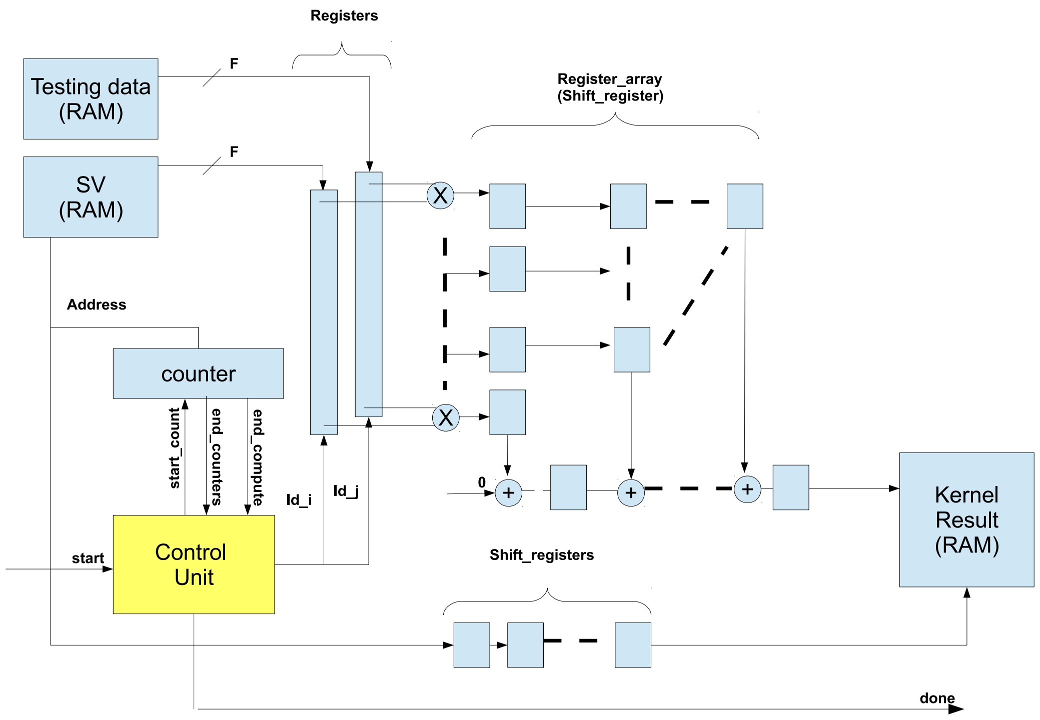


Fig. 1: Linear kernel proposed hardware architecture

internal FPGA block RAM to simplify data access. The testing data RAM stores all the test data, whereas the SV RAM stores all support vectors. All support vectors are concatenated with each other to maximize the utilization of each memory block. In this implementation, features data width are parameterizable to cater different precision requirement by different types of application.

The number of feature determines the number of multipliers needed for the dot product computation. In this work, we focus on linear kernel since its the basis to any kernel computation. Based on eqs.(4) to (7), each of kernel computation needs a dot product computation. The architecture shown in Fig. 1 can be expanded to any kernel implementation. The addition process is done in pipelined to reduce the critical path delay. The number of features also determines the number of adders that determines the initial latency of proposed architecture.

The resultant output is stored in an on-chip memory. However, depending on the application requirement in case the result is to be used for the next computation immediately, the shift register can be replaced to produce a valid output signal. The done signal is to determine the end of the computation. Shift registers are placed to synchronize the memory access to the input and output RAMs. Hence, only one address counter is needed for this implementation.

V. Result and Discussion

The proposed architecture has been evaluated with different number of features and support vectors. As previously mentioned, SVM configurations for different applications differ based on these parameters. Most implementations of SVM on FPGA were benchmarked for performance based on one or more datasets. In Kane et al. [13], nine datasets were used to benchmark their proposed architecture in comparison to CPU and GPU. Each dataset has unique number of support vectors, features and classes. In order to have a detailed hardware performance analysis of the linear kernel architecture, our experimental focuses only on varying the number of features and support vectors. The number of classes is not taken into account since it is not computed at the kernel stage.

Our analysis has been carried in two scenarios. For experiment 1, the number of support vectors is fixed to 32. The number of features is adjusted from 5 onwards with increment of 5 until the device hits its maximum resource utilization. For experiment 2, the number of features is fixed to 20. The number of support vectors are adjusted from 2 onwards, doubled its prior value, until the device hits its maximum capacity. The data width of each support vectors, feature and testing data are fixed to 32 bits.

The increment of one address bit at support vector RAM (refer Fig. 1) would result in double the memory size. For both experiments the proposed architecture is analyzed in terms of logic resources, embedded multipliers, memory and maximum operating frequency. By changing a certain parameter while keeping the rest constant, we are able to analyze how a particular parameter affects the performance of the linear kernel architecture.

We used Quartus II v13.0 to emulate the system. The linear kernel has been modeled in Verilog and the RTL simulation in Quartus has been done using ModelSim 6.6 to obtain cycle accurate results. An identical software implementation is done in MATLAB to verify our results. The system implementation was done on Cyclone IV EP4CE115F29C7 device. In our targeted device, there are 114,480 logic elements, 3.98 Mbits of internal memory and 529 9-bits embedded multipliers.

Fig. 2 shows the performance of linear kernel implementation as the number of features increases. As the number of features increases the logic unit, embedded multiplier and memory utilization increase. When all embedded multipliers are used up, logic resource utilization increases at a much faster rate since more logic elements are needed to implement additional multipliers. The maximum operating frequency remains consistent for <70 features since the architecture is fully pipelined. For >70 features there is a drop in frequency this is due to increase size of adders and multipliers to handle values wider data width.

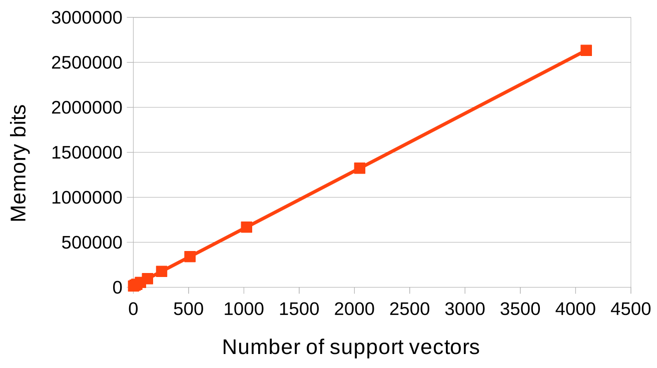


Fig. 3: The impact of the number of support vectors on memory bits utilization of the linear kernel architecture

Fig. 3 shows the performance of linear kernel when then number of support vectors increases. In this linear kernel architecture, support vectors only affects the on-chip memory utilization. Hence it is vital for support vectors to be stored on-chip since each test data needs to be multiplied with all of the support vectors. Off-chips support vectors access would greatly affect the performance of the system. The logic elements (4380 – 4520), embedded multipliers (160) and maximum operating frequency (82 – 87 MHz) were not significantly affected with the number of support vectors.

Based on eq. (8), each training data input needs to be multiplied with each support vector. Therefore, the number of clock cycles needed to process the training data is equivalent ot the number of support vectors. The maximum throughput this architecture is $\frac{1}{N_{SV}}$. If the input data rate is greater than the maximum throughput, a FIFO is required to buffer incoming input.

VI. Conclusion

In this paper, we proposed the architecture for linear kernel architecture on FPGA. We analyzed the hardware requirement for different number of of support vectors and also the number of features. From our proposed linear kernel implementation, the number of features determine the maximum operating frequency and logic resource utilization, whereas the number of support vector determines the amount of on-chip memory usage and also the overall kernel throughput. Therefore, application dataset with larger number of support vectors would result in lower throughputs. In future work, we target on implementing the full SVM classification architecture and analyze the impact of features, support vectors and number of class towards the hardware performance.

Acknowledgment

This work is supported in part by the Collaborative Research in Engineering, Science & Technology (CREST) grant P17C114 (vote no. 4B176) and Universiti Teknologi Malaysia Matching grant (vote no. 00M75).

References

[1] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, and H. P. Graf, “A massively parallel FPGA-based coprocessor for support vector machines,” in *Field Programmable Custom Computing Machines, 2009. FCCM’09. 17th IEEE Symposium on*. IEEE, 2009, pp. 115–122.

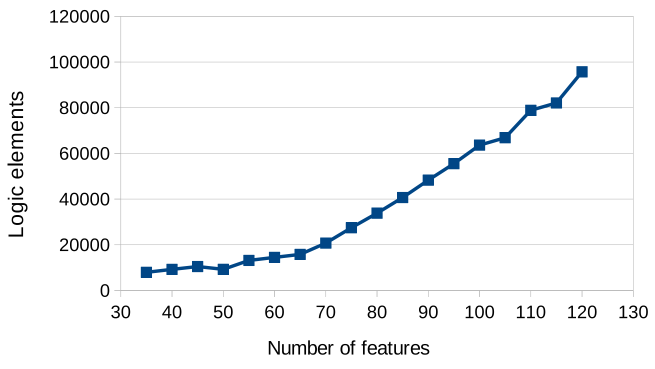
[2] M. P. Véstias, “High-performance reconfigurable computing granularity,” *Encyclopedia of Information Science and Technology*, pp. 3558–3567, 2015.

[3] S. B. Kotsiantis, “Supervised machine learning: A review of classification techniques,” in *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007, pp. 3–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=1566770.1566773

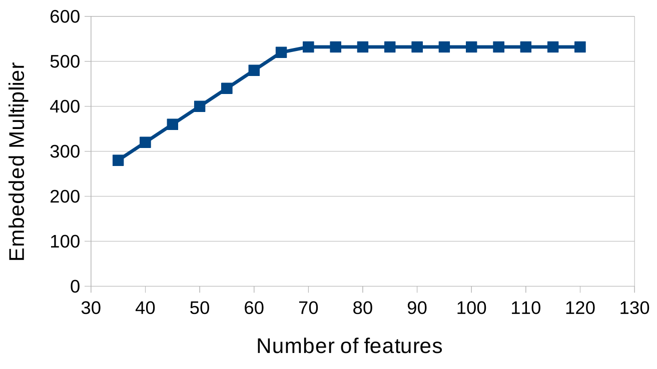
[4] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[5] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, “FPGA-based real-time pedestrian detection on high-resolution images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 629–635.

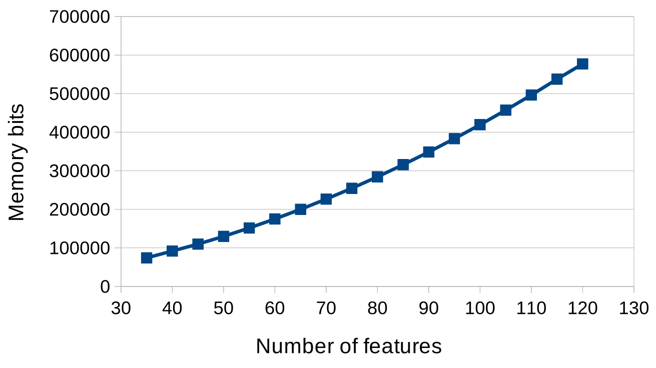
[6] C. Kyrkou, C.-S. Bouganis, T. Theocharides, and M. M. Polycarpou, “Embedded hardware-efficient real-time classification with cascade support vector machines,” *IEEE transactions on neural networks and learning systems*, vol. 27, no. 1, pp. 99–112, 2016.



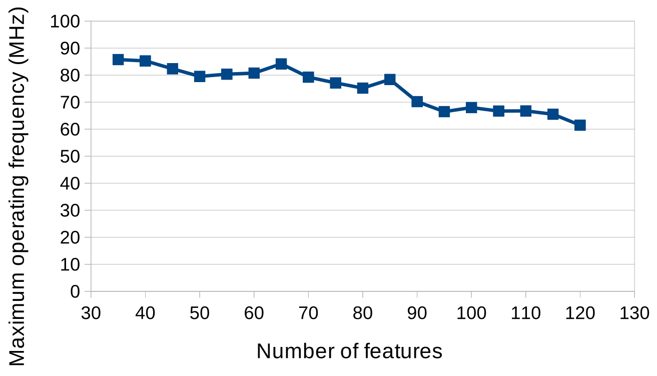
(a) Logic elements versus number of features



(b) Embedded Multiplier versus number of features



(c) Memory bits versus number of features



(d) Maximum operating frequency versus number of features

Fig. 2: The impact of the number of features on the performance of the linear kernel architecture

[7] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "An FPGA implementation of a HOG-based object detection processor," *IPSJ Transactions on System LSI Design Methodology*, vol. 6, no. 0, pp. 42–51, 2013.

[8] C.-P. Shen, W.-C. Kao, Y.-Y. Yang, M.-C. Hsu, Y.-T. Wu, and F. Lai, "Detection of cardiac arrhythmia in electrocardiograms using adaptive feature extraction and modified support vector machines," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7845–7852, 2012.

[9] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "FPGA implementation of real-time head-shoulder detection using local binary patterns, SVM and foreground object detection," in *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*. IEEE, 2012, pp. 1–8.

[10] L. Bustio-Martínez, R. Cumplido, J. Hernández-Palancar, and C. Feregrino-Uribe, "On the design of a hardware-software architecture for acceleration of SVMs training phase," in *Mexican Conference on Pattern Recognition*. Springer, 2010, pp. 281–290.

[11] J.-F. Wang, J.-S. Peng, J.-C. Wang, P.-C. Lin, and T.-W. Kuan, "Hardware/software co-design for fast-trainable speaker identification system based on smo," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1621–1625.

[12] S. Venkateshan, A. Patel, and K. Varghese, "Hybrid working set algorithm for SVM learning with a kernel coprocessor on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2221–2232, 2015.

[13] J. Kane, R. Hernandez, and Q. Yang, "A reconfigurable multiclass support vector machine architecture for real-time embedded systems classification," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*. IEEE, 2015, pp. 244–251.

[14] O. Chapelle, B. Schölkopf, and A. Zien, "Introduction to semi-supervised learning," 2006.

[15] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.

[16] X. Song, H. Wang, and L. Wang, "FPGA implementation of a support vector machine based classification system and its potential application in smart grid," in *Information Technology: New Generations (ITNG), 2014 11th International Conference on*. IEEE, 2014, pp. 397–402.

[17] K.-k. Cao, H.-b. Shen, and H.-f. Chen, "A parallel and scalable digital architecture for training support vector machines," *Journal of Zhejiang University SCIENCE C*, vol. 11, no. 8, pp. 620–628, 2010.

[18] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: traps in big data analysis," *Science*, vol. 343, no. 6176, pp. 1203–1205, 2014.

[19] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural computation*, vol. 13, no. 3, pp. 637–649, 2001.

[20] M. B. Rabieah and C.-S. Bouganis, "FPGA based nonlinear support vector machine training using an ensemble learning," in *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*. IEEE, 2015, pp. 1–4.

[21] S. Wang, Y. Peng, G. Zhao, and X. Peng, "Accelerating on-line training of LS-SVM with run-time reconfiguration," in *Field-Programmable Technology (FPT), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[22] M. Papadonikolakis and C.-S. Bouganis, "Novel cascade FPGA accelerator for support vector machines classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1040–1052, 2012.

[23] M. Ruiz-Llata, G. Guarnizo, and M. Yébenes-Calvino, "FPGA implementation of a support vector machine for classification and regression," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–5.